Signal Delay in General RC Networks with Application to Timing Simulation of Digital Integrated Circuits*

Tzu-Mu Lin and Carver A. Mead

Abstract—Modeling digital MOS circuits by RC networks has become a well-accepted practice for estimating delays. In 1981, Penfield and Rubinstein proposed a method to bound the waveforms of nodes in an RC tree network. In this paper, a single value of delay is derived for any node in a general RC network. The effects of parallel connections and stored charges are properly taken into consideration. The algorithms can be used either as a stand-alone simulator, or as a front end for producing initial waveforms for waveform relaxation-based circuit simulators. An experimental simulator called SDS (Signal Delay Simulator) has been developed. For all the examples tested so far, this simulator runs two to three orders of magnitude faster than SPICE, and detects all transitions and glitches at approximately the correct time.

1 Introduction

Modeling digital MOS circuits by RC networks [9], [11] has become a well-accepted practice for estimating delays. In 1981, Penfield and Rubinstein (P-R) proposed a method to bound the waveforms of nodes in an RC tree network [11]. This method is conceptually simple and computationally efficient, and has been incorporated into many timing-analysis programs [6].

Two approximations are made in the P-R method: 1) modeling the input of transistors by step waveforms, and 2) modeling conducting transistors by linear resistors. Later, Horowitz (H) extended this method to include both effects of slow inputs and nonlinearity of MOS transistors [4], [5]. In addition to the bounds of a waveform, he also derived a single time constant estimate of the waveform. Summarized in the following are some observations on the results of his work:

1. In most (linear) RC networks, the waveform of an output node can be approximated by a single exponential function with time constant equal to the Elmore's delay [3] of the node. Another property of this approximation is that the integral of the difference between the approximated waveform and the real waveform is zero.

2. If nonlinearity of MOS transistors is considered, then output waveforms can no longer by approximated by exponential functions. However, if only delay values are of interest, not detailed waveforms, then Elmore's delays are still applicable. The only complication is to use different (effective) resistance values for rising and falling transitions of signals.

*Re-typeset from original material by Donna Fox, June 2017. Originally published in *1984 Conference on Advanced Research in VLSI,* MIT, January 24, 1984.

3. Although the situation is more complicated if the slow input effect is also included, the delay of an output node can still be approximated by a function of Elmore's delays. In this case, however, the delay of an output node also depends on the delays of input nodes (outputs of previous stages).

What is implied in the above observations is that, as far as delay is concerned, the nonlinear behavior of MOS circuits can be absorbed in the effective resistances of transistors. The dependency of delays on network topologies can be determined using the original linear formulation proposed by Elmore [3]. The great usefulness of this formulation lies in its analysis and composition capabilities.

One deficiency of the work of P-R-H is that Elmore's delay is only derived for RC tree networks, not for general RC meshes. Furthermore, the effect of initial charge is only considered for a special case that an RC tree without any initial charge is driven through another RC tree that is fully charged initially. No generalization is made to deal with networks with arbitrary initial charge distributions. The purpose of this paper is to derive Elmore's delay for general RC networks with parallel (and bridge) connections and any initial charge distribution. The emphasis is on algorithms for calculating delays, and on applications of these algorithms to timing simulations of digital MOS circuits. Details of the theory can be found in [8].

2 Composition of Elmore's Delay

Prior to any discussion, the following remarks are noted first [5], [8].

1. In this context, the term RC network refers only to those networks that are approximations of MOS circuits, i.e., resistor networks where there is a capacitor between every node and GND.

2. An RC network is assumed to be driven by one and only one source (VDD or GND) which is referred to as the source of the network.

3. The measure the delay of a node in an RC network, it suffices to consider the normalized case where the node voltage starts from some initial value between 0 and 1, and is driven towards the final value 1. The results obtained in this normalized case are easily adapted to both charging and discharging processes, and to any value of supply voltages. Normalized variables are used throughout the context; that is, V is dimensionless and therefore Q is of the same dimension as C.

The original definition of Elmore's delay is

$$T_D^0 = \int_0^\infty t y'(t) dt. \tag{1}$$

where $y'(t)$ is the derivative of the waveform $y(t)$ of some node of a linear network. The superscript ^{0} indicates zero initial charge, the condition always assumed by Elmore (also by P-R). This definition has been applied to derive analytic expressions of delays in various types of linear networks. It is also one of the five parameters used for deriving the P-R bounds. A modification of this definition is necessary here because the original formulation (Eq. 1) only makes sense when $y(t)$ is monotonic. In an RC tree network without any initial charge, the waveform of any node is guaranteed to be monotonic [12]; however, monotonicity is not true in general. To deal with general RC networks, the delay is redefined as

$$T_D = \int_0^\infty 1 - y(t) dt. \tag{2}$$

This expression is just the area above $y(t)$, but below 1. In case of zero initial charge, T_D reduces to T_D^0 [8], [12]. Using this definition of delay, a two-port RC network [8], [11] is characterized by three parameters:

1. R (series resistance): the resistance between the input port and the output port of the network,

2. $\overline{C}$ (effective capacitance): $\overline{C} = C - Q$, where C is the total capacitance, and Q is the total charge in the network,

3. D (internal delay): the delay of the output port when the output port is open and the input port is directly driven by the source.

These three parameters are directly related to the coefficients of the transmission matrix of the two-port RC network [8]. As two-port RC networks are composed in various ways, these three parameters can be calculated in a hierarchical manner. The composition rules agree with those described in [11], except that stored charge is properly taken into consideration. We also add composition rules for parallel connections. These rules are presented as follows:

1. primitive case (a resistor in series with a capacitor. The other end of the capacitor is connected to GND):

$$\begin{aligned} R &= r \\ \overline{C} &= c(1 - v_0) \\ D &= rc(1 - v_0) \end{aligned} \tag{3}$$

where r, c and v_0 are the values of the resistance, the capacitance, and the initial voltage of the capacitor, respectively.

In the following three cases, a subscript is associated with each parameter, indicating to which network this parameter belongs. In particular, subscript $_T$ indicates the resulting network of each composition.

2. series connection of N_1 and N_2:

$$\begin{aligned} R_T &= R_1 + R_2 \\ \overline{C}_T &= \overline{C}_1 + \overline{C}_2 \\ D_T &= D_1 + D_2 + R_1\overline{C}_2 \end{aligned} \tag{4}$$

3. N_S with side branch N_L:

$$\begin{aligned} R_T &= R_S \\ \overline{C}_T &= \overline{C}_L + \overline{C}_S \\ D_T &= D_S + R_S\overline{C}_L \end{aligned} \tag{5}$$

4. parallel connection of $N_1, \cdots, {}_n$:

$$\begin{aligned} R_T &= \frac{1}{\sum_1^n \frac{1}{R_i}} \\ \overline{C}_T &= \sum_1^n \overline{C}_i \\ D_T &= R_T \left(\sum_1^n \frac{D_i}{R_i} \right) \end{aligned} \tag{6}$$

Finally, the delay of a node in an RC network can be expressed in terms of the $(R, \overline{C}, D)$ parameters of the driving and loading networks of the node. Let D_S and R_S be the D and R parameters of the driving network, and $\overline{C}_L$ be the $\overline{C}$ parameter of the loading network. Then, the delay of the node equals

$$D_S + R_S \overline{C}_L \tag{7}$$

2.1 RC Tree Network

From Eq. 7, the delay of a node depends on both the driving and loading networks of the node. Parallel (and bridge) connections couple all nodes together so that every node is driving and loading every other node at the same time. As a result, the calculation of delays in general needs to be carried out independently for each individual node. However, no node in a tree network both drives and loads another node, so the delays of all the nodes can be calculated simultaneously and incrementally. The following algorithm (TREE) calculates the delays of all the nodes in a tree network.

1. The loading information is accumulated and propagated from the loading ends towards the driving end of the tree network. To be more precise, a value C_i^L is associated with each node i, and

$$C_i^L = \begin{cases} \frac{\overline{C}_i}{C_i} + \sum_j C_j^L & \text{if node } i \text{ is a leaf} \\ & \text{otherwise} \end{cases}$$

where index j ranges over all the succeeding nodes of node i, and $\overline{C}_i$ is the effective capacitance of node i.

2. The delay of each node is calculated incrementally from the driving end towards the loading ends, i.e.,

$$T_i = T_{p(i)} + r_i C_i^L \tag{8}$$

where $p(i)$ is the parent node of node i, and r_i is the resistance between node i and node $p(i)$.

The time complexity of this algorithm is $\mathcal{O}(n)$, where n is the number of nodes in the tree network.

Another formula that expresses the delay of a node i in an RC tree network, due to P-R [11], is

$$\sum_k R_{i,k} \overline{C}_k \tag{9}$$

where $R_{i,k}$ is the resistance of the (unique) path between the source and node i, that is in common with the (unique) path between the source and the node k. $\overline{C_k}$ is the effective capacitance of node k. The summation carries over all nodes k in the network.

With Eq. 6 and Eq. 7, the delay of any node in an RC network can be calculated. This process is direct, constructive, but requires information regarding the global topology of the network. Presented in the next section is another approach of delay calculation that is iterative and distributive in nature [2]. Each node or transistor is itself a process, which only communicates with its neighboring nodes and transistors. The delays of all the nodes are determined in a collective manner. This approach is capable of dealing with general RC networks without sacrificing the desirable property of tree networks described above.

3 Tree Decomposition and Load Redistribution

The problem of evaluating delays can be reformulated as a set of relations among neighboring nodes and branches (transistors). Associate a global index with each node. Suppose there are a_i branches incident on a node N_i. Let $r_{(i,j)}$ denote the resistance of the jth branch, and $f(i,j)$ denote the global index of the neighboring node of N_i through this branch. The idea here is to split $\overline{C}_i$ into these a_i branches, each of value $\overline{C}_{(i,j)}$, such that $\sum_{j=1,\cdots,a_i} \overline{C}_{(i,j)} = \overline{C}_i$, and the delays evaluated from different branches are the same. $\overline{C}_{(i,j)}$ is the equivalent load to node N_i from the jth branch. By Eq. 8, $T_i = T_{f(i,j)} + r_{(i,j)}\overline{C}_{(i,j)}$. To summarize, we have the following set of relations

$$\begin{cases} T_i = T_{f(i,j)} + r_{(i,j)}\overline{C}_{(i,j)} & j = 1, \cdots, a_i \\ \sum_{j=1}^{a_i} \overline{C}_{(i,j)} = \overline{C}_i & i = 1, \cdots, N \end{cases} \tag{10}$$

where N is the number of nodes in the network. Eq. 10 represents a system of linear equations: $r_{(i,j)}$'s and $\overline{C}_i$'s are known, and T_i's and $\overline{C}_{(i,j)}$'s are to be determined. Note that $C_{(i,j)}$'s may either be positive or negative. If it is a negative number, then the jth branch is driving, not loading node N_i. We do not intend to solve Eq. 10 directly because of the enormous number of variables involved: $\sum_{i=1}^{N}(a_i + 1)$. Note that the a_i branches incident on node N_i need not be decoupled completely as we did in the formulation of Eq. 10. These branches can be divided into any number ($b_i, 1 \leq b_i \leq a_i$) of groups. Rather than fully decoupled into nodes and transistors, the network is decomposed into a smaller number of subnetworks. Delays are calculated directly and independently inside each subnetwork using the techniques discussed in previous sections. The consistency of the delay of a common node shared by more than one subnetwork is checked and corrected by a procedure similar to the formulation of Eq. 10. As delays can be calculated very efficiently for a tree network, we require that all decomposed subnetworks be trees. The root of every tree must be the source of the RC network.

There are two steps involved in decomposing an RC network into trees. The first step is purely topological, while the second step concerns the distribution of node capacitances, as well. The first step is referred to as "tree composition," and the second step is referred to as "load distribution." For a given RC network, tree decompositions always exist, and based upon the concept of dominant paths [1], one such decomposition scheme is presented in the next section. The discussion in the present section applies to any tree decomposition of RC networks.

Given a tree decomposition of an RC network, the system of linear equations (Eq. 10) can be reduced to another system of linear equations consisting of only $\sum_{u=1}^{P}(b_i - 1)$ variables, where P is the number of nodes that split, and each of these nodes split into b_i parts, $i = 1, \cdots, P$, respectively. Eq. 9 is the main formula used in this reduction process. The matrix associated with this reduced system of linear equations is symmetric and positive-definite [8]. If the (point or block) Gauss-Seidel method is used to solve this system, then convergence is guaranteed [14]. In fact, a more practical algorithm exists to solve this reduced system of linear equations. This algorithm only uses information from neighboring nodes and transistors, and under certain conditions [8], it is equivalent to the block Gauss-Seidel method. The algorithm is as follows. Given any initial load distribution for a tree decomposition of an RC network, the delay of each node is calculated using algorithm TREE. The iteration process starts by scanning through those nodes that are split, and checking if the delay evaluated from different branches are the same. If they are not, node capacitances are distributed improperly somewhere in the network. Although nothing is known as to where this improper distribution happens, one can always adjust the local distribution of $\overline{C}_{(i,j)}$'s so that the delays evaluated from different branches are equal for the node presently under investigation. The adjustment is done as follows. Suppose N_i is the current node under investigation, and the delay evaluated from different branches $T_{(i,1)}, \cdots, t_{(i,b_i)}$ are not all equal. Based upon the fourth rule

of Eq. 6, the delay of node N_i is given by

$$T_i = \frac{\sum_{j=1}^{b_i} \frac{T_{(i,j)}}{R_{(i,j)}}}{\sum_{j=1}^{b_i} \frac{1}{R_{(i,j)}}}. \tag{11}$$

where $R_{(i,j)}$ is the source resistance of node $N_{(i,j)}$, and which remains fixed during the iteration process. Let $\Delta C_{(i,j)}$ be the amount of load adjustment for node $N_{(i,j)}$. Then

$$\Delta C_{(i,j)} = \frac{T_i - T_{(i,j)}}{R_{(i,j)}} \tag{12}$$

The constraint $\sum_{j=1}^{b_i} \overline{C}_{(i,j)} = \overline{C}_i$ is satisfied automatically since

$$\sum_{j=1}^{b_i} \Delta C_{(i,j)} = \sum_{j=1}^{b_i} \frac{T_i - T_{(i,j)}}{R_{(i,j)}} = T_i \sum_{j=1}^{b_i} \frac{1}{R_{(i,j)}} - \sum_{j=1}^{b_i} \frac{T_{(i,j)}}{R_{(i,j)}} = 0. \tag{13}$$

To maintain consistency, this adjustment of $\overline{C}_{(i,j)}$'s must be propagated to other nodes in the same tree so that their delays can be updated accordingly. Instead of updating the delays of all nodes in the same tree when there is a change of load, a more efficient approach is taken to accumulate the changes as the scan process goes along. The delay of a node is not updated until it is scanned. Instead of scanning through the nodes in the original network, the nodes in the decomposed network are visited in a depth-first manner. This algorithm, called LRD (Load ReDistribution), is described in the following pigeon code.

```
procedure LRD;
  var source=secondary_node;
      "source of the network"
 function scan(N:secondary_node; T0:delay)
  =capacitance;
  var∑T=delay; ∑C,cl=capacitance;
     S=secondary_node;
  begin
     "N.primary:corresponding primary node"
     "N.sons:succeeding nodes"
     "N.R:source resistance"
     "N.ΔC:  Eq. 12"
     "N.T:delay"
      N.T:=N.T+T0;
      combine(N.primary);"Eq. 11 & Eq. 12"
      if N.sons = nil then scan:=N.ΔC
      else begin
        ∑T:=T0+N.ΔT*N.R;
        ∑C:=0.0;
        for S ∈ N.sons do begin
           c1:=scan(S,∑T+N.R*S.c2);
           ∑C:=∑C+c1;
           S.c2:=∑C;
           ∑T:=∑T+c1*N.R;
```

```
          N.T:=N.T+c1*N.R;
        end;
        for S ∈ a.sons
        do S.c2:=∑C-S.c2;
        scan:=N.ΔC + ∑C;
      end;
  end;(* scan *)
begin (* LRD *)
  while not converge do
  for S ∈ source.sons do scan(S,0.0);
end; (* LRD *)
```

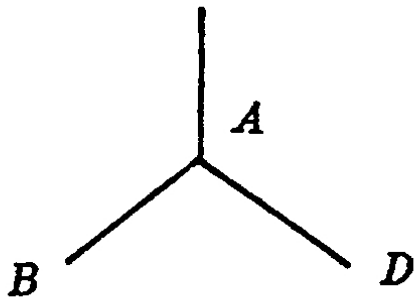

Figure 1: Idea Behind Procedure "Scan."

A "primary node" refers to a node of the original network. After decomposition, it splits into a number of "secondary nodes." The idea behind procedure "scan" is indicated by the following relationship among the three nodes A, B and D of Fig. 1:

1. $\Delta T_B|A, D(\equiv \Delta T_B|_\Delta C_i = 0, i \neq A, D) = R_{B,A}\Delta C_A + R_{B,D}\Delta C_D = R_A(\Delta C_A + \Delta C_D) = \Delta T_A|A, D.$

2. $\Delta T_A|B, D = R_{A,B}\Delta C_B + R_{A,D}\Delta C_D = R_A(\Delta C_B + \Delta C_D).$

Note that $R_{A,B}$ is defined in Eq. 9. The first equation above suggests the accumulation and propagation of Δ_T (parameter $T0$ of procedure "scan") from the driving end towards the loading ends. The second equation suggests the accumulation of Δ_C (returned by procedure "scan") from the loading ends towards the driving end. If branch B is scanned before branch D, then $\Delta T_D|B$ is in effect at the present scan. On the other hand, the value of ΔC_D is stored at variable B.c2 to update T_B when node B is scanned at the next iteration step.

The proof of correctness and examples of using this algorithm are given in [8]. The time complexity is $\mathcal{O}(l \cdot Q)$, where l is the number of iteration steps used, and $Q = \sum_{i=1,B_i\neq 1}^{N} b_i = \sum_{i=1}^{P} b_i$. The number of iteration steps required depends on the accuracy aimed at. Usually, four or five steps are enough to bring the error down to 10 percent.

4 Application to Timing Simulation

The TREE and LRD algorithms have been applied in an experimental timing simulator called SDS (Signal Delay Simulator). The simulation model is based upon Bryant's switch-level logic simulation model [1]. In this model, transistors are modeled as resistances, and nodes are modeled as capacitances. With each node is associated one of three different states corresponding to the node voltage: 1 (high voltage), 0 (low voltage) and X (in transition). A transistor may be either on or off depending on the state of the node controlling its gate. The evolution of a MOS circuit is approximated by a sequence of RC networks. Various node capacitances are charged to VDD and discharged to GND through the network. The time when a node changes state is determined by the delay value of the node. When the gate node of at least one transistor changes state, a new network results. A partially charged or discharged node which connects to the gate of a transistor does not change the state of that transistor. However, the charge stored in the nodes will be taken into account when the nodes are again charged or discharged through the new network.

The concept of dominant paths [1] is used not only to determine logic states, but also to decompose network into trees for estimating delays. The initial load distribution is made such that a node only loads those nodes that are along its dominant paths, and has no effect on the nodes on other paths. This approximation always decomposes a network into a collection of trees since a node A cannot be in the dominant path of another node B if node B is in the dominant path of node A. If a node has more than one dominant path, then the load is equally distributed among these paths. In most cases, the delay estimated under this approximation is already very accurate. For other cases, algorithm LRD is used to calculate the exact delay of every node. The mechanism for scheduling logic events according to delays is quite similar to that presented by Terman [13]. As each new event is evoked, logic states, delay values, and stored charge of affected nodes are updated accordingly.

The per-square resistances for different types and usages of transistors [6] are calibrated using SPICE [10]. Moreover, as mentioned in the introduction of this paper, different resistance values are used for rising and falling of signals.

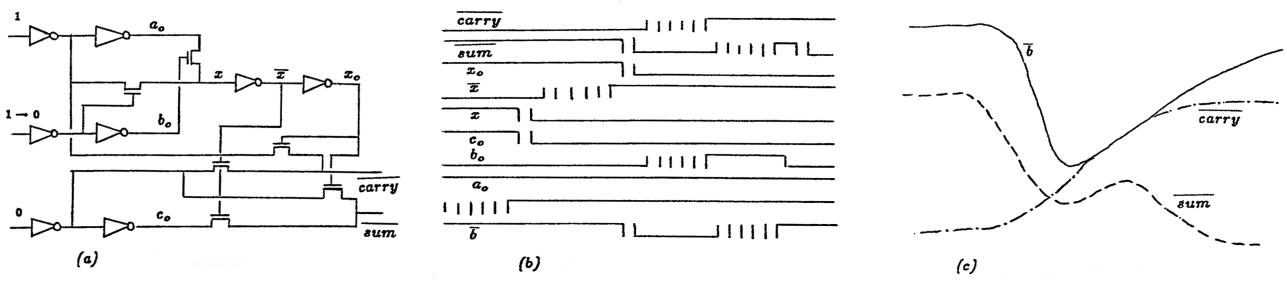

Figure 2: Comparison between SDS and SPICE. (a) An NMOS one-bit adder; (b) SDS simulation result; (c) SPICE simulation result.

Shown in Fig. 2(a) is an NMOS one-bit full adder used in a student project at Caltech. The simulation results generated by SPICE for the case that the three input bits change from (1,0,1) to (1,0,0) is shown in Fig. 2(b). Simple as it is, this example serves as a good test case of SDS because there are many feedback and multiplexed paths in the circuit. The simulation result produced by SDS is shown in Fig. 2(c). The time intervals of some transitions and glitches estimated by SDS are compared with the waveforms generated by SPICE. All the glitches and transitions are detected at approximately the correct time. Both programs run on a DEC-20 computer. SPICE takes about 40 sec (CPU), and SDS takes about 0.15 sec (CPU). The difference is two to three orders of magnitude, which is typical for the examples tested so far.

Recently, a new circuit simulation technique called "waveform relaxation" (WR) has been reported in the literature [7]. This technique is claimed to have nice numerical properties, and can speed up circuit analysis by an order of magnitude or more over SPICE. One possible application of the algorithms presented in this paper is to provide initial waveforms for WR-based circuit simulators. Note that a good initial guess of waveforms is crucial to the performance of this type of circuit simulators.

5 Conclusions

The area criterion of Eq. 2 is used throughout this paper as the definition of delay. For any RC network driven by a single source, the delay value of any node can be determined precisely. The effects of parallel connections and stored charges are properly taken into consideration. As an application, an experimental simulator called SDS has been developed. For all the examples tested so far, this simulator runs about two to three orders of magnitude faster than SPICE, and detects all the transitions and glitches at approximately the correct time. The algorithms can be used either as a stand-alone simulator, or as a front end for producing initial waveforms for waveform relaxation-based circuit simulators. The generalization of the simulation model and delay-calculation algorithms to high-level representations of networks is presently under investigation.

References

[1] Bryant, R.E., "A Switch-level Simulation Model for Integrated Logic Circuits," Doctoral Dissertation, MIT/LCS/TR-259, MIT (March, 1981).

[2] Chen, M.C., "Space-Time Algorithms: Semantics and Methodology," Doctoral Dissertation, Computer Science, Caltech, 5090:TR:83 (May, 1983).

[3] Elmore, W.C., "The Transient Response of Damped Linear Networks with Particular Regard to Wideband Amplifiers," Journal of Applied Physics, Vol. 19, No.1, pp. 55–63 (January, 1948).

[4] Horowitz, M., "Timing Models for MOS Pass Networks," International Symposium on Circuits and Systems, pp. 198–201 (1983).

[5] Horowitz, M. "Performance Modeling for MOS Integrated Circuits," Doctoral Dissertation, Stanford University (October, 1983).

[6] Jouppi, N.P., "TV: An nMOS Timing Analyzer," Proc. of the 3rd Caltech VLSI Conference, pp. 71–86 (March, 1983).

[7] Lelarasmee, E., Ruehli, A.E., and Sangiovanni-Vincentelli, A.L., "The Waveform Relaxation Method for Time-Domain Analysis of Large-Scale Integrated Circuits," IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, CAD-1, No.3, pp. 131–145 (July, 1982).

[8] Lin, T-M. and Mead, C.A., "Signal Delay in General RC Networks with Application to Timing Simulation of Digital Integrated Circuits," Computer Science, Caltech, 5089:TR:83 (1983).

[9] Mead, C.A. and Conway, L.A., "Introduction to VLSI Systems," Chapter 1, Addison-Wesley (1980).

[10] Nagel, L. and Pederson, D., "Simulation Program with Integrated Circuit Emphasis (SPICE)," Proc. of the 16th Midwest Symposium on Circuit Theory (April, 1973).

[11] Penfield, P. and Rubinstein, J., "Signal Delay in RC Tree Networks," Proc. of the 2nd Caltech VLSI Conference, pp. 269–283 (March, 1981).

[12] Penfield, P., Rubinstein, J., and Horowitz, M., "Signal Delays in RC Tree Networks," IEEE Trans. on Computer-Aided Design (1983).

[13] Terman, C.J., "Simulation Tools for Digital LSI Design," MIT/LCS (December, 1981).

[14] Varga, R.S., "Matrix Iterative Analysis," Prentice-Hall Series in Automatic Computation (1962).